On this page

# Routing

## Configuring Routes

Routes are configured by rendering `<Routes>` and `<Route>` that couple URL segments to UI elements.

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router";
import App from "./app";

const root = document.getElementById("root");

ReactDOM.createRoot(root).render(
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<App />} />
    </Routes>
  </BrowserRouter>
);
```

Here's a larger sample config:

```
<Routes>
  <Route index element={<Home />} />
  <Route path="about" element={<About />} />

  <Route element={<AuthLayout />}>
    <Route path="login" element={<Login />} />
    <Route path="register" element={<Register />} />
  </Route>

  <Route path="concerts">
    <Route index element={<ConcertsHome />} />
```

```
      <Route path=":city" element={<City />} />
      <Route path="trending" element={<Trending />} />
    </Route>
  </Routes>
```

## Nested Routes

Routes can be nested inside parent routes.

```
<Routes>
  <Route path="dashboard" element={<Dashboard />}>
    <Route index element={<Home />} />
    <Route path="settings" element={<Settings />} />
  </Route>
</Routes>
```

The path of the parent is automatically included in the child, so this config creates both `"/dashboard"` and `"/dashboard/settings"` URLs.

Child routes are rendered through the `<Outlet/>` in the parent route.

app/dashboard.tsx

```
import { Outlet } from "react-router";

export default function Dashboard() {
  return (
    <div>
      <h1>Dashboard</h1>
      {/* will either be <Home/> or <Settings/> */}
      <Outlet />
    </div>
  );
}
```

## Layout Routes

Routes *without* a `path` create new nesting for their children, but they don't add any segments to the URL.

```
<Routes>
  <Route element={<MarketingLayout />}>
```

```
      <Route index element={<MarketingHome />} />
      <Route path="contact" element={<Contact />} />
    </Route>

    <Route path="projects">
      <Route index element={<ProjectsHome />} />
      <Route element={<ProjectsLayout />}>
        <Route path=":pid" element={<Project />} />
        <Route path=":pid/edit" element={<EditProject />} />
      </Route>
    </Route>
  </Routes>
```

## Index Routes

Index routes render into their parent's `<Outlet/>` at their parent's URL (like a default child route). They are configured with the `index` prop:

```
  <Routes>
    <Route path="/" element={<Root />}>
      {/* renders into the outlet in <Root> at "/" */}
      <Route index element={<Home />} />

      <Route path="dashboard" element={<Dashboard />}>
        {/* renders into the outlet in <Dashboard> at "/dashboard" */}
        <Route index element={<DashboardHome />} />
        <Route path="settings" element={<Settings />} />
      </Route>
    </Route>
  </Routes>
```

Note that index routes can't have children. If you're expecting that behavior, you probably want a layout route.

## Route Prefixes

A `<Route path>` *without* an `element` prop adds a path prefix to its child routes, without introducing a parent layout.

app/routes.ts

```
  <Route path="projects">
```

```
    <Route index element={<ProjectsHome />} />
    <Route element={<ProjectsLayout />}>
      <Route path=":pid" element={<Project />} />
      <Route path=":pid/edit" element={<EditProject />} />
    </Route>
  </Route>
```

## Dynamic Segments

latest API Reference

### Routing

```
<Route path="teams/:teamId" element={<Team />} />
```

app/team.tsx

```
import { useParams } from "react-router";

export default function Team() {
  let params = useParams();
  // params.teamId
}
```

You can have multiple dynamic segments in one route path:

```
<Route
  path="/c/:categoryId/p/:productId"
  element={<Product />}
/>
```

app/category-product.tsx

```
import { useParams } from "react-router";

export default function Team() {
  let { categoryId, productId } = useParams();
  // ...
}
```

You should ensure that all dynamic segments in a given path are unique. Otherwise, as the `params` object is populated - latter dynamic segment values will override earlier values.

## Optional Segments

You can make a route segment optional by adding a `?` to the end of the segment.

```
<Route path=":lang?/categories" element={<Categories />} />
```

You can have optional static segments, too:

```
<Route path="users/:userId/edit?" component={<User />} />
```

## Splats

Also known as "catchall" and "star" segments. If a route path pattern ends with `/*` then it will match any characters following the `/`, including other `/` characters.

```
<Route path="files/*" element={<File />} />
```

```
let params = useParams();
// params["*"] will contain the remaining URL after files/
let filePath = params["*"];
```

You can destructure the `*`, you just have to assign it a new name. A common name is `splat`:

```
let { "*": splat } = useParams();
```

## Linking

Link to routes from your UI with `Link` and `NavLink`

```
import { NavLink, Link } from "react-router";

function Header() {
  return (
    <nav>
      {/* NavLink makes it easy to show active states */}
      <NavLink
```

```
          to="/"
          className={({ isActive }) =>
            isActive ? "active" : ""
          }
        >
          Home
        </NavLink>

        <Link to="/concerts/salt-lake-city">Concerts</Link>
      </nav>
    );
  }
```